**UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE**

**FAKULTA PRÍRODNÝCH VIED A INF**

**Vizualizácia rôznych typov grafov a algoritmov z teórie grafov**

**BAKALÁRSKA PRÁCA**

**Nitra 2022 Lukáš Hajda**

**UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE**

**FAKULTA PRÍRODNÝCH VIED inf**

**Vizualizácia rôznych typov grafov a algoritmov z teórie grafov**

**BAKALÁRSKA PRÁCA**

**ASPON 40 STRAN**

**Nitra 2022 Lukáš Hajda**

**Obsah**

Úvod…………………………………………………………………………………………………

1. Analyza sucasneho stavu => popis nieco o praktickom vyuziti grafov
2. Ciele prace
3. Teoria grafov ……………………………………………………………………………………
   1. Základné princípy a definície ……………………………………………………………..
      1. Grafy ……………………………………………………………………………
      2. Grafová súvislosť a komponenty…………………………………………..
      3. Najkratšia cesta ……………………………………………………………….
      4. Stromy a lesy…………………………………………………………….
      5. Kostra grafov …………………………………………………………………..
   2. Algoritmy …………………………………………………………………..
      1. Prehľadávanie do hĺbky ………………………………………………………
      2. Prehľadávanie do šírky ………………………………………………………
      3. Bellman-Ford algoritmus …………………………………………………….
      4. Kruskalov algoritmus …………………………………………………………
      5. Primov algoritmus……………………………………………………………
      6. Tarjanov algoritmus ………………………………………………….
      7. Topologické usporiadanie ……………………………………………………

1.3 Reprezentacia grafov......................................................................................................

1. Postup vytvarania webovej aplikacie …………………………………………………………………….

2.1 Cytoscape kniznica ……………………………………………………………………….

2.2 Implementacia algoritmov………………………………………………………………..

2.3 Uzivatelske programove rozsirenia …………………………………………………….

2.3.1 Grafová reprezentácia pre dva programovacie jazyky.........……………….. Záver ……………………………………………………………………………………………….

Zoznam bibliografických odkazov………………………………………………………………..

Zoznam príloh ………………………………………………………………………………………

**Zdroje**

1. Dokumentacia k Cytoscape kniznici: <https://js.cytoscape.org/>
2. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: Introduction to algorithm Third Edition
3. Nečas Jiří: Grafy a jejich použití, 1978
4. Kolář Josef, Štěpánková Olga, Chytil Michal: Logika, algebry a grafy, 1989
5. J.Matoušek a J.Nešetril: Kapitoly z Diskretnej Matematiky, 2007
6. J. Černý: Základní grafové algoritmy, 2010
7. M.J.Bannister, D.Eppstein: Randomized speedup of the Bellman-Ford Algorithm, 2011
8. M.Sharir: A Strong-connectivity algorithm and its applications in data flow analysis, 1980
9. S.Palúch: Algoritmická teória grafov, 2008
10. P.Hlinený: Základy teorie grafu, 2010
11. J.Pócs: **Grafy a Sítě II, 2021**
12. Mark Needham & Amy E. Hodler : Graph Algorithms Practical Examples in Apache Spark & Neo4, 2019
13. Rob Landley, 2007, <https://docs.kernel.org/core-api/rbtree.html?highlight=red>
14. <https://edu.ukf.sk/pluginfile.php/239742/mod_resource/content/0/DBS_prednasky_07.pdf>
15. C. J. Cheneyhttps://people.cs.umass.edu/~emery/classes/cmpsci691s-fall2004/papers/p677-cheney.pdf
16. Graph Algorithms, 2nd Edition, SHIMON EVEN
17. **https://is.muni.cz/el/fi/jaro2019/IB002/um/IB002-2019-slajdy.pdf** Černá , 2019
18. <https://www.ijert.org/research/applications-of-depth-first-search-a-survey-IJERTV2IS70557.pdf> Gaurav Rathi, Dr. Shivani Goel
19. <http://diplom.utc.sk/wan/1275.pdf> DIPLOMOVÁ PRÁCA Realizácia simulačného modelu distribuovanej kontroly topológie v Ad-Hoc sieťach Marek Čipka
20. <https://math.fel.cvut.cz/en/people/demlova/lgr/p-lgr815.pdf> Marie Demlová: Logika a grafy
21. https://watermark.silverchair.com/btv557.pdf?token=AQECAHi208BE49Ooan9kkhW\_Ercy7Dm3ZL\_9Cf3qfKAc485ysgAAAt0wggLZBgkqhkiG9w0BBwagggLKMIICxgIBADCCAr8GCSqGSIb3DQEHATAeBglghkgBZQMEAS4wEQQM4RtQifClKEK2RLWhAgEQgIICkBPRQzfEMQ42bdee60bG017B5SmNuCisSu-LaSrYHfTMWcXd8sS-TCfr2o-LrjaVYBEZbFJEbZ9Jy8xopwUGypKmRQDvy5LjlojPBj320KKSqYqJIWDeJQZRyXjmjCYjpZs-wTt-7mqkVZwiO5\_BP8IsgKzpo-oT3Jn-1Ibsk02M5uJ-vZfRpUUVm\_65kn5RwUCJTzA7gLP6mxYHPbpLujxpf1coZOCAAJTY7MhRnAgxa1P6icaf6pCWER4LWaqxrsqnkKwNp5OT5-KOu\_sKhXPcXnPE\_RQHfHL8a0NKsNlMR4KlY3Nrah\_kcrTsxpNKjSh9K2GBXa4jyzrPui1ksUa6Lio8-zhiWSE9oaw0uYGbMkJ035BLIQsjwSwolx90zP7DzVAMX7wwpT6qO02mmqCe3LUOVCNTK9XhD0JC6FJgL\_8PRb-NxCl-EQ-xHiMvKpT2EEkvpwZIQLsxgTrWJ2MJWrZWtWv1KHrxsIdCPzBU3Aw9vuOmn6dtl2pLjseOQTfm-Nv04W3NO0BORL1xr2e79u9u3zMFT9RVewakLKEUKmmIBHaoSRRX0DhPebN29VgeGI9C7TlXL5n11TRP9xVNRpXpFca15C2rBwaCyP51rYr\_NGrcT1pYFqoYSXMg-SdvwGllPE3RGGiZm0HNcKmb6Nw4fSlRapASt\_04HZshllIA0fKbFN1TzzSyN3AMFTagoMOTbB\_xmqwtaGrtv-ZotNgZ3syWw5jq\_a6CSkvZe8ov5b7ZQX\_VIDCD1XvjwB07ksYzCf6eUOyRSUTq9fI0yo9RCGVHz7ipSL69rhYK4B0dAl4ud4JHiyBkfiOSviFEQNdpAWpU3k2H0UEE7UxVAeNxB8ukIDPG4CsRDn7s [Franz, Lopes, Huck, Dong, Sumer ,Bader, 2015]

V predkladanej bakalarskej praci je riesenia …..

## **Anotácia:**

Doplnit: garbage collcetor

Teória grafov a problémy, na ktoré ponúka riešenia formou grafových algoritmov, majú široké uplatnenie v našom živote. V niektorých prípadoch sa bez týchto riešení vieme zaobísť, ale v niektorých, napr. pri hľadaní najbližšej cesty k cieľu, optimálnej trasy pri rozvoze tovaru alebo implementácií dnes už viac obľúbených kryptomien sa stáva ich uplatnenie nutnosťou.

Ciele práce: vizualizácia vybraných typov grafov aj s možnosťou ich aplikovania pri vizualizácii vybraných algoritmov najčastejšie používaných v teórii grafov na praktické účely. Práca bude obsahovať popis fungovania a podstaty jednotlivých algoritmo, možnosti ich aplikácie v praxis a programátorské rozširenie.

Charakter práce: vývoj softvéru.

Požiadavky na obsah podľa charakteru práce: popis riešeného problému, návrh softvéru, metodika vývoja, implementácia, popis vytvoreného softvéru, testovanie.

Požiadavky na vedomosti a zručnosti študenta:

analytické, abstraktné myslenie, zručnosť v programovaní webových stránok (Javascript, HTML, CSS a pod.), záujem o problematiku algoritmov z teórie grafov.

Grafy

Velke mnozstvo matematickych, informatickych a taktiez praktickych problemov vieme vyjdrati pomocout abstraktneho objektu pozostavajuci z dvoch veci:

* (konecne) mnozina bodov (vrcholov)
* mnozinu vzajomnych vztahov (hran)

Body mozu reprezentovat napriklad mesta na mape a hrany zas cesty medzi nimi. Alebo body mozu vyznacovat osobu na socialnych sietach a pomocou hran vieme zistit kto s kym je “priatel” a kto zas nie. Tento sposob reprezentacie ponima matematicky objekt nazyvany **graf**. Tento objekt bol prvy krat pouzity vyznamym matematikom Leonhardom Eulerom na riesenie problem siedmych mostov v roku 1736 avsak disciplina zaoberajucaja sa grafmi vznikla v prvej polovoci dvadsiateho storocia. Neorientovany Graf *G* definujeme ako usporiadanu dvojicu (V, E), kde V je neprazdna mnozina vrcholov a E je mnozina ( hran ) dvojprvkovych podmnozin mnoziny V. Hranu medzi vrcholmi *u* a *v* definujeme ako dvojprvokovu mnozinu {u, v} alebo skratene *uv*.

**Orientovany graf (digraf):**

Orientovany graf G je usporiadana dvojica (V, E) kde E V x V [Hlineny]

**Uplny graf:**

Graf G = (V, E) sa nazyva uplny prave vtedy ked kazde dva jeho (navzjaom rozne) vrcholy su spojene aspon jednou hranou teda E = {(u, v) : u, v V }

Graf G = (V, H) nazveme úplným, ak množina H obsahuje všetky možné dvojice typu {u, v}, kde u, v ∈ V a u 6= v. [Paluch]

**Sled:**

Sled v grafe G = (V, E) je lubovolna striedava postupnost hran a vrcholov v tvare:

µ(v1, vk) = (v1, {v1, v2}, v2, {v2, v3}, v3, . . . , {vk−1, vk}, vk). [Paluch]

**Ťah:**

Je taky sled v grafe G = (V, E), v ktrom sa ziadna hrana neopakuje [Paluch]

**Cyklus:**

Je uzavretý ťah v ktorom sa okrem prvého a posledného vrchola žiaden vrchol nevyskytuje viac než raz.[Paluch]

**Podgraf:**

Graf G` = (V`, E`) je podgrafom grafu G = (V, E) prave vtedy, ked ` a ` teda `. [Paluch] [definicia podgrafu]

**Faktor grafu:**

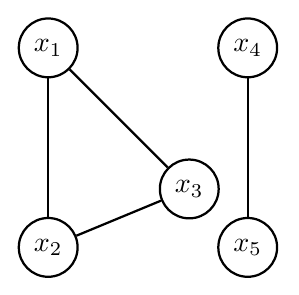
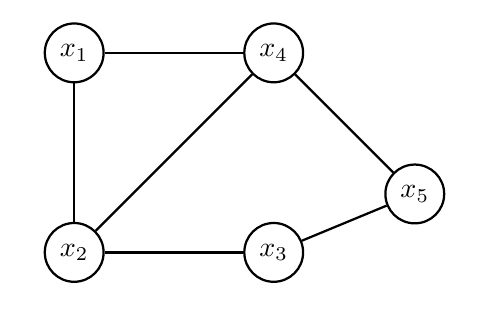
Graf G`=(V`, E`) je faktor grafu G = (V, E) prave vtedy ked V` = V tzv. graf G` obsahuje vsetky vrcholy grafu G. [Necas 1978]

Grafova suvislost a komponenty:

Ako sme uz vyssie spominali grafmi dokážeme namodelovat urcitu realnu část nasho života jako napr. Dopravy systém pomocou ktoreho sa dokážeme dostat z jedneho mesta do druhého. Je celkom pochopitelne že v tomto systéme chceme mat moznost sa dostat z každého miesta do ineho. Při neorientovanych grafov kde kazdy jeden vrchol je dosiahnutelny z ineho vrchola nazýváme aj **suvisly inak nesuvisly graf**. Při orientovanych grafoch dalej definujeme pojmy **silne suvisly** a **slabo suvisly** graf.

Def:  
Graf G = (V, E) nazyvame suvisly ak pre kazdu dvojicu vrcholov (*u, v*) e *V existuje (v, u) cesta. Inak hovorime ze graf G* je nesuvisly.

Doplnit definiciu SLABO A SIONO SUVISLY

Silna komeponenta:

Original:

“”Silný komponent digrafu D je maximálny silný poddigraf digrafu D, t.j. silný poddigraf taký, že žiadny jeho naddigraf v rámci D už nie je silný. Ak D1, D2, . . . , Dt sú všetky silné komponenty digrafu D, tak V (D1) ∪ V (D2) ∪ · · · ∪ V (Dt) = V (D) (pripomíname, že digraf s jedným vrcholom je silný). Naviac musí platiť V (Di) ∩ V (Dj ) = ∅ pre všetky i 6= j, ináč by všetky vrcholy z množiny V (Di) ∪ V (Dj ) boli navzájom dosažiteľné jeden z druhého a teda vrcholy z V (Di) ∪ V (Dj ) by patrili do rovnakého silného komponentu”” [Jozef Pócs]

Moje upravene:

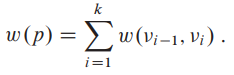
Silný komponent orientovaneo graf G = (V, E) maximálny silný podgraf grafu G = (V, E). Ak G`1, G`2, . . . , G`t sú všetky silné komponenty orientovaneho grafu G tak V (G`1) ∪ V (G`2) ∪ · · · ∪ V (G`t) = V (G) (pripomíname, že orientovany graf s jedným vrcholom je silný). Naviac musí platiť V (G`i) ∩ V (G`j ) = ∅ pre všetky i != j, ináč by všetky vrcholy z množiny V (G`i) ∪ V (G`j ) boli navzájom dosažiteľné jeden z druhého a teda vrcholy z V (G`i) ∪ V (G`j ) by patrili do rovnakého silného komponentu

K1, K2 a K3 su silne komponenty daného orientovaného grafu.

Najkratsia cesta

Hladanie najkratsej cesty medzi dvoma (alebo viacerymi) vrcholmi v grafe je jedna z najzakladnjesich algoritmických uloh v treorii grrafov ktora ma svoje prakticke vyuzitie jako napr. Hladanie najkratsej cesty z města A do města B, dorucenie paketov cez internetovu siet ku koncovému prijimatelovi apod. Medzi jedny z najznamejsich algoritmov ktore vyhladavaju najrkatsiu cestu v ohodnotenych orientovanych grafov je Dijsktrov algoritmus, ktory avsak pre svoje spravne fungovanie potřebuje kladne ohodnotene hrany. S negativne ohodnotenymi hranami funguje algoritmus Bellmana a Forda ktory je oproti Dijskrovemu o nieco pomalsi.

V orientovanom grafe G = (V, E) definujeme vahu *w(p)* cesty p = <*v0, v1, ..,vk*> jako sucet vah hran ktorymi cesta prechadza.

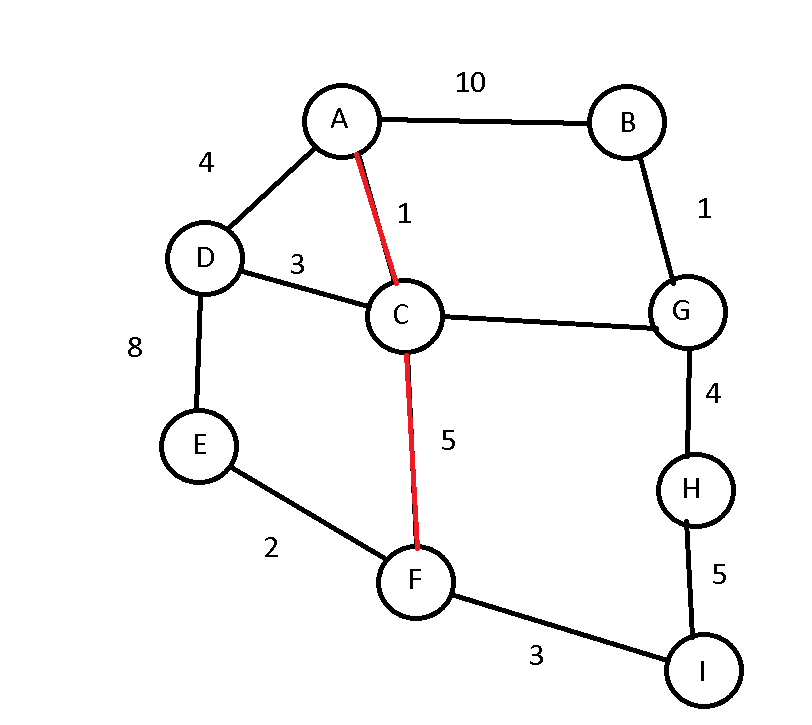


Dalej definujeme **vahu najkratsej cesty** δ(u, v) z vrcholu *u* do vrcholu *v*jako:

Text

Description automatically generated

Potom **najakrasia cesta** v ohodnotenom orientovanom grafe G = (V, E) je zadefinovana ako akakolvek cesta *p* s vahou *w(p) =* δ(u, v)*.*

[Pocs, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein]

Najkratsia cesta z vrcholu A do vrcholu F je ma vahu 6 a cesta je: A-C-F

**Strom a les + obrazok**

Strom v informatike je často pouzivana datova struktura ktorej rozne modifikacie nam dovoluju rychlo ukládat a vyhladavat data. Medzi najznamejsie modifikacie stromov patri napr. B-strom ktory je pouzivany v databazach, dalej pozname cerveno-cierny strom ktory sa pouziva v kerneli operacneho systemu Linux ale ani v teorii grafov tento pojem neni neznamy. Strom v teorii grafov reprezentuje velmi jednoduchy graf ale napriek tomu tvoria zaujimavu oblast studia.

S**trom**

1. Suvisly grafm ktory neobsahuje kruznicu [kole, stepankova, chytil 1989]
2. Suvisly acyklicky graf

[obrazok stromu]

**Les**

Graf bez kruznic, jednotlive kompotenty grafu su stromy.

[obrazok slesu]

**Kostra grafu**

V teorii grafov kostra *K* je podgraf neorientovaného suvisleho grafu *G,* ktoru získáme postupnym odstranovanim hran tak aby sme nenarusili suvislost grafu a aby medzi každými dvoma vrcholmi existovala prave jedna cesta. Nečas (1978) definuje kostru grafu tak ze ak G je neorientovany suvisly graf, K je jeho faktor a ak K je zaroven stromom, hovorim ze K je kostra grafu G.

V ohodnotenych neorientovanych grafov rozlisujeme najdrahsiu a najlacnejsiu kostru. Paluch (2008) uvadza najdrahsiu kostru v grafe G s najvacsou cenou a najlacnejsiu kostru v grafe G zas s najmensou cenou kde Cena c(K) kostry K je sucet ohodnoteni jej hran. 10

[příloha kostra grafu: ohodnoteny]

Algoritmy

**Prehladavanie do sirky:**

Prehladavanie do sirky (Breadth-first search) uz iba BFS, patri medzi jeden z najzakladnejsich a najlahsich prehladavacich algoritmov ktory sluzi jako zakladny kamen pre komplexnejsie algoritmy ako je napr. Primov algoritmus pre hladanie minimalnej kostry grafu.

**Postup:**

Tento algoritmus prehladva vrcholy tak ze najprv odhali vsetkych nasledovnikov pociatocneho vrcholu ale s tym ze si pamatam v akom poradi ich ohalil, toto zapamatanie mozme z programtorskeho hladiska zaistiti pouzitim datovej struktury nazyvanej fronta do ktorej postupne vkladame odhalene vrcholy a ked su vsetci nasledovnici odhaleni tak zo zaciatky fronty si vybereme další vrchol a algoritmus opakujeme. Toto odhalovanie skonci ak su vsetky dosazitelne vrcholy odhalene, inac povedane ked fronta bude prazda. Algoritmus dokaze upravit aj tak aby sa odhalil cely graf a nie len z pociatocneho vrcholu ().

**Atributy vrcholov:**

Vrcholy obsahuju nasledujuce atributy :

* *v.pi –* vrchol z ktoreho bol vrchol *v* objaveny (predchodca). Na zaciatku maju vsetky vrchol nastavene na *Nil*
* *v.d –*  vrstva *,* vramci ktorej bol dany vrchol v odhaleny. Při inicializacii maju vsetky vrcholy nastavene na *Nekonecno,* pociatocny vrchol ma 0.
* *v.color –*  Seda farba reprezentuje ze dany vrchol bol objaveny, Biela farba zre vrchol nebol este objaveny a Ciernu farbu ma vrchol uz bol preskumany.

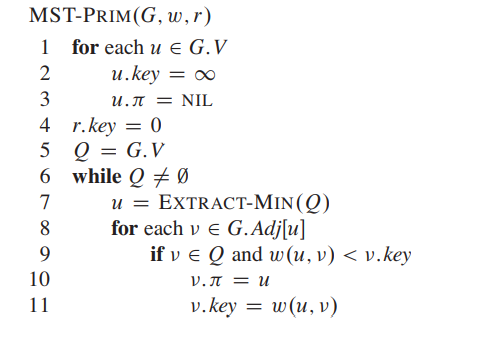
**Zlozitost:**

Asymptoticka zlozitost BFS algoritmu je O(V + E) (kde V je množina vrcholov a E je mnzoina hran) lebo v najhorsom pripade bude muset prejst kazdou jednou hranou a kazdym vrcholom grafu.

**Vyuzitie:**

BFS algoritmus ma rozne vyuzitie v realnom svete jedno z nich je aplikacia v GPS systemoch na hladanie najkratsej cesty. Dalej sa vyuziva vo web crawleroch pre lepsie zaindexovanie stranok a v neposlednom rade ho dokonca vyuziva jeden algoritmus pre garbage collection metodu.

**Pseudo kod**



Obrazok 1 pseudocode (autor, rok)

**Prehladavanie do hlbky**

Prehladavanie do hlbky (Depth-first search) dalej uz iba DFS, je dalším prehladavacim algoritmom, ktoreho nazov nam evokuje ze graf budeme prehladavat „hlbsie“ dokial sa bude dat.

**Postup**:

DFS algoritmus objavuje hrany ktore vychadzaju z naposledy odhaleného vrcholu *v.* Ked algoritmus objavi koncovy vrchol ktory este nebol odhaleny, oznaci ho za objaveny. Ak vsetky hrany z naposledny odhaleného vrcholu *v* boli objavene tak „backtracking“ metodou sa vrati naspat do vrcholu z ktoreho bol vrchol *v* objaveny a algoritmus sa opakuje pokial sa neodhalia vsetky dosazitelne vrcholy z pociatocneho vrcholu.

**Atributy vrcholov:**

Vrcholy obsahuju nasledujuce atributy:

* *v.pi -* vrchol z ktoreho bol vrchol *v* objaveny (predchodca). Na zaciatku maju vsetky Nil
* *v.d* a *v.f –* casove znacky kde *v.d* udava kedy bol vrchol odhaleny a *v.f* zas kedy bol vrchol cely preskumany.
* *v.color* – Biela kde este nebol odhaleny, Seda kde je odhaleny ale este neskončilo odhalovanie jeho nasledovnikov a Cierna ked prehladavanie vrcholu *v*skončilo.

**Zlozitost:**

Asymptoticka zlozitost DFS algoritmu je O(V + E).

**Vyuzitie:**

DFS mozme vyuzit napr při generovani bludiska.Taktiez sluzi jako sucast algoritmu pre hladanie silnych komponent (Tarjanov algoritmus). Dalsim vyuzitim je hladanie cyklu v grafe a v neposlednom rade sluzi na hladanie topologickeho usporiadania.

[odkaz na pseudocode]

**Algoritmus Bellmana a Forda**

BFS a DFS algoritmy fungovali na neohonodtenych grafoch ale modelovat realny svet bez ohodnotenych hran by bolo privelmi tazke ba az nemozne. Existuje niekolko algoritmov, ktore hladaju najkratsie cestu v ohodnotenych orientovanych grafoch ale v tejto bakalraskej praci sme sa rozhodli popisat algoritmus Bellmana a Forda ktory sice je oproti Dijskrovemu algoritmu pomalsi avšak dokaze pracovat so zápornými hraniami a lepsie detekovat negativne cykly.

**Postup:**

Bellman a Ford algoritmus hlada najkratsiu cestu z pociatocneho vrcholu do vsetkych ostatanych vrcholov. Majme orientovny ohodnoteny graf G = (V, E) s pociatanocnum vrcholom *s a* funckiu *w: E 🡪 R.* Algoritmus vracia booleanovsku hodnotu ktora nam hovori ci existuje negatvny cyklus ktory je dosiahnutelny z pociatocneho vrcholu. Ak taky cyklus neexituje algoritmus vytvořit najkratsie cesty a vrati hodnotu True.

Algoritmus vyuziva metodu relaxaciu hran ktorou postupne znizuje hodnotu *v.d*  vsetkych vrcholov *v e V* na hodnotu najkratsej cesty z pociatocneho vrcholu az pokial sa nedosiahne najkratsia cesta d(s, v). [algrotimy a datove struktury, rok]

**Atributy vrcholov:**

* *v.pi –* vrchol z ktoreho bol vrchol *v* objaveny (predchodca). Na zaciatku maju vsetky vrchol nastavene na *Nil*
* *v.d –*  vzdialenost (distance)

**Zlozitost:**

Aymptoticka zlozitost algoritmu je O(V . E) ale při urcitom tpe grafu dokážeme tuto zlozitost znizit. Při orientoavanych acyklických grafoch si dokazem zistit topologicke usporiadanie a podla tohoto usporiadania postupne relaxovat hrany v takom poradi akom sa vyskytuju na ceste. Takto dokazem znizit asymptoticku zlozitsto algoritmu na O(V + E).

**Vyuzitie:**

Rozne modifiakcie algoritmu Bellmana a Forda sa vyuzivaju napr v smerovacom protokole **RIP**.

[BellmanFord pseudocode]

**Kruskalov algoritmus**

Vstupny grafmoze obsahovat vela minimálních kostier a najst tu najlepsiu z nich sa moze zdat ako obtiazna uloha avsak tento problém je algoritmicky velmi dobře zvladnutelny pomocou Kruskalovho algoritmu ktory sa radi medzi „pazrave“ algoritmy. Tieto algoritmy sa vyznacuju tym ze vyberaju vramci aktualneho stavu vždy tu nejlepsiu moznost.

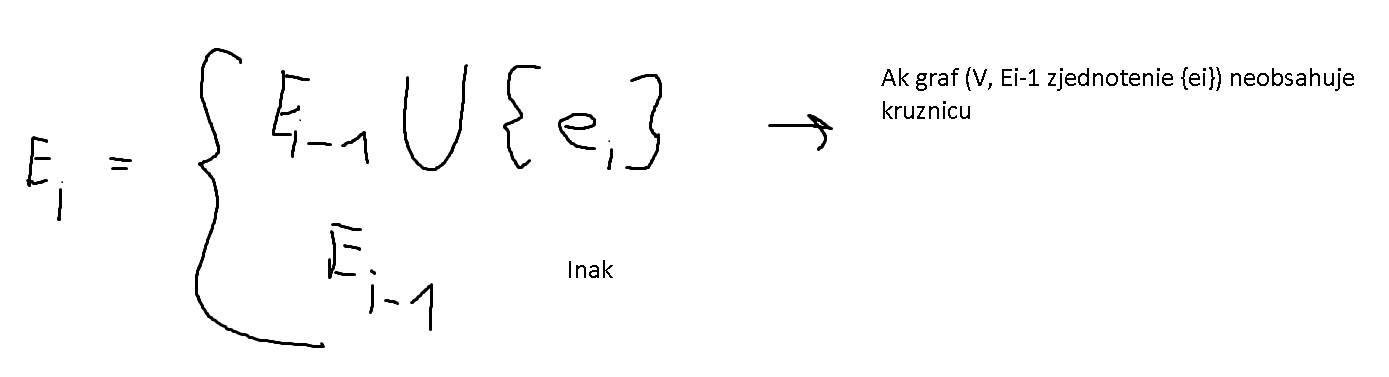
**Postup**

**M**atousek a Nesetril (2007) opisuju krusaklov algoritmus nasledovne:

Mame suvisly **ohodnoteny** graf G = (V, E) s n vrcholmi, m hranami a s odhonotenim w. Ocislujeme si hrany e1, e2,…em **zoradime** tak aby

w(e1) <= w(e2) <= … <- w(em).

Nasledne budeme postupne konstruovar mnoziny hran E0, E1, … E . Polozime E0 = Ø. Ak bola uz najdena množina Ei-1 tak mnozinu Ei spočítáme nasledujuco:

[]

**Zlozitost**

Zlozitost

Algoritmus sa zastavi ak Ei obsahuje n-1 hranu aleb osa spracovali vsetky hranu grafu G.

Majme suvisly ohodnoteny neorientovany graf G = (V, E) na ktorom potrebujme najst minimalnu kostru *K.* Ocislujeme si hrany e1, e2…em s ohodnodetim

Hrany grafu G si usporiadame vzostupne podla ich ohodnoteni *w*. Nasledne jako uvadzaju autori Matousek, Nesetril (2007) budeme postupne konstruovat mnoziny hran E0, E1, E2 …. E. Polozime E0 = Ø. Ak bola uz najdena mnozna Ei-1, spočítáme množinu Ei:

**Primov algoritmus**